

The Tentun project

David Gordon and Laurent Birtz
Dépt. de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke, Québec, Canada J1K 2R1
email : birtl00@dm.usherb.ca gordd00@dm.usherb.ca

Abstract

1 introduction

The Tentun project is primarily dedicated to the exploration of security risks introduced by the tunneling concept.

Tunneling is the concept of hiding a message into another one. Over the years, different forms of tunneling have been implemented. For instance, tunneling IP traffic over the X.25 packet layer protocol was common practice in the early days of the Internet.

Today, tunneling has many uses. For example, SSH uses a form of tunneling to establish a secure channel between two parties. Tunneling is also used to establish virtual private networks.

Many tools aim at the establishment of tunnels. Vtun, Fragroute, HttpTunnel and IcmpChat are good examples of tunneling tools. With the proliferation of such tools, tunneling is increasingly a security issue. Indeed, by hiding information into another packet, illicit conversations can be established that no firewall can detect. This is a big problem because information can be stolen without any traces.

Tentun is a tunneling project that introduces the innovative concept of modifying existing packet traffic to include completely transparent covert traffic. Our primary goal is to show that undetectable illicit conversations can be established between two parties and, eventually, we hope to develop firewalling techniques against these intrusions.

To show the power of tunneling, we have, in a first experimentation, implemented a socket to socket tunnel over a network. The tunnel itself is simply raw UDP or TCP between the sockets. The challenge is to become familiar with the resources mentioned earlier and do system network changes programmatically.

In our second experimentation, we used protocol data space to hide a data stream of a different protocol. For example, our implementation would be able

to handle a FTP connection by encoding all FTP traffic within a HTTP connection. For now, the HTTP connection would be a generated one, not an actual legitimate HTTP connection.

Our final experiment involves stretching the limits of protocol specifications. For instance, a normal NTP stream of data would be modified to include a covert telnet stream. The modification could be to insert the telnet stream into the TCP options field of the NTP packets.

As we just presented, tunneling has been implemented in many different tools in a fragmented manner. The Tentun project also aims to unify tunneling implementations as well as to introduce a completely innovative concept. Instead of using protocol data space, Tentun intends to bend protocol RFC specifications to create covert streams of data in existing data streams.

In section 1, we present existing tunneling tools and implementations. In sections 2, 3 and 4, we present the results of our experimentation. Section 5 presents our unifying approach. Finally, we conclude by presenting some future directions for this work.

2 Tunneling Concept

Traditionally, tunneling is the transmission of data intended for use only within a private network. The most common use of tunneling is by far to create VPNs. Tunneling is generally done by encapsulating the private network data and protocol information within public network packets so that the private network protocol information appears to the public network as data.

Technically, tunneling is simply the process of putting one packet into another. Yet, tunneling is not just an encapsulation process. If it was the case, the fact of integrating a HTTP packet into an IP packet would be a tunnel. This kind of encapsulation occurs in any network environment where protocols

are implemented as layers (ISO and TCP/IP).

Tunneling is more often used to put a packet into another that lies at the same level of the protocol hierarchy. As an example, tunneling is used in mobile IP to encapsulate an IP packet into another IP packet. Other examples are the PPTP and L2TP protocols, and the IPSec protocol which create tunnels to provide secure channels.

Technically, a tunnel works the following way:

1. The original message is sent through the normal channel;
2. The original message is intercepted;
3. The original message is modified to produce a new packet (t-packet);
4. The t-packet is sent through the network to the other end of the tunnel;
5. The t-packet is received and the original packet is reconstructed;
6. The original packet is sent to its original destination.

It should be noted that in order for any tunnel to work, the tunneling software must be present both at the source and destination nodes.

As an example of tunnel, let's consider a company, having multiple branches in different cities, that want a simple way to send reports throughout the entire company. The company provides all its employees a local posting service for all its branches. To send a report to a employee in another branch, one must put it in an envelop of the local post services and just put it in the local post. When the local post service receives a letter for another branch, it puts it in an envelop provided by the public post services and sends it through this service. When the mail arrives to the other branch, it is removed from the public postal envelop and put back in the local post services.

Now, let's consider the same example but applied to networking: creating an IPv6 tunnel in an IPv4 Internet. In figure 1, there are three networks. The source and destination IPv6 networks are separated by a IPv4 network (Internet). The solution is to encapsulate IPv6 packets within an IPv4 packet. The tunnel exists between two routers that have both IPv4 and IPv6 addresses.

The important aspects of tunneling are:

1. How is the original message intercepted?

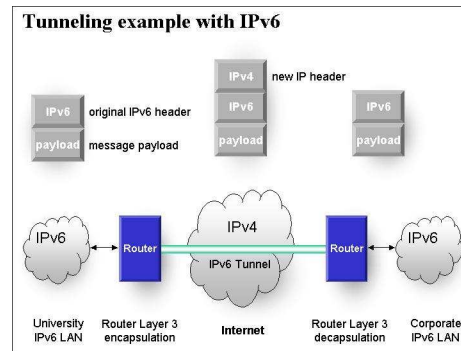


Figure 1: Figure 1 - IPv6 tunneling

2. How is the original message modified ?
3. What is the scope of the tunnel?

2.1 Packet interception

There are multiple ways of intercepting a packet once it is sent. This can be done locally or remotely. The tunneling client/server are responsible for re-sending the intercepted packet. The general idea is that packet interception will not lead to duplication of the packet in the network. This will be termed active packet interception.

2.1.1 TCP Port Forwarding

The simplest form of packet interception is TCP port forwarding similar to SSH tunnels. A tunnel is established between a client and a server on a given source port, say 1000, and a destination port, 80. All traffic output on server side will be forwarded to the correct port (and host). For example, a telnet connection might be made to local port 1000 (client side tunnel), forwarded to port 80 (server side tunnel), then forwarded to port 23 (telnet server).

The disadvantage of this approach is that only protocols above the TCP layer can be supported.

2.1.2 Loopback Sniffing

This approach has been used by Dug Song in his tool fragroute. In order to create a tunnel for a particular destination, he will modify the routing table to send all packets with that destination to the loopback. Of course, the packets will die there, but the tunneling application is sniffing on the loopback. It rewrites the packets using a raw socket (writing directly to ethernet), thus bypassing the routing table. The packets

are sent to the destination and are intercepted by a tunneling server application.

The advantage of this approach is that the entire IP stack is tunneled, therefore all protocols are effectively supported. One disadvantage is that the tunneling server has no way of knowing which packets to intercept.

2.1.3 Device Assisted Tunneling

The approach adopted by VTun is to use a tunneling device. The routing table can be modified so that all packets for a certain destination are sent to the tunneling device (ie. tun0 on Linux), instead of the ethernet device (ie. eth0). In kernel mode, the tunneling device returns the packets in user mode, so that the tunneling application can perform its magic. Once packets arrive at the remote host, they are intercepted by the tunneling server. The server will decapsulate them and send them through the tunneling device back into kernel mode.

Similar to loopback sniffing, the advantage of this approach is that the entire IP stack is tunneled. This approach is also standard in most operating systems.

2.1.4 Remote interception

To intercept the packet remotely, a pair of gateways between the source and destination must be configured to act as tunneling agents. Naturally, for the tunnel to function properly, all communication must be redirected through the gateways. The first gateway will intercept a packet and introduce it into the tunnel. The second gateway will receive the tunneled packet and extract the original packet to send to the destination. Methods for doing this can be as explained above or by simply applying a filter based on destination address and port for instance.

2.1.5 Passive interception

For tunneling purposes, passive packet interception is also possible. However, this introduces the problems of having two copies of the packet in the network, a modified and an original copy. One solution is to modify certain packet fields so that the operating system will reject the modified packet as invalid. A problem with this approach is that an IDS will detect that half the packets are rejected. However, it is possible to fool the IDS by using a method similar to an insertion attack. This is not part of the scope of the article, it is mentioned as a possibility to explore.

2.2 Packet modification

We see two ways to modify messages. The first way is to insert our message to be tunneled into a fake message that serves as a transport. The second is to use an existing message as a transport for our message.

Using this knowledge, two types of tunnel are distinguished by the Tentun project. The first type of tunnel is one that uses the data space of the tunneling protocol to send data. The second type of tunnel is an innovation in tunneling that Tentun introduces, which involves creating a tunnel out of existing messages using packet fields other than data spaces.

2.2.1 Protocol Data Space Tunnel

A tunnel that uses the data space of a tunneling protocol to send its data is the standard in the industry nowadays. In other words, a tunneling protocol specifically expects to be sending a packet of another protocol within itself. The original packet is encapsulated in the tunneling packet and sent to the other endpoint of the tunnel where it is decapsulated. Once decapsulated, the original packet can be processed normally (forwarded to the correct host or port). Usually, the encapsulated packet will be encrypted. However, this type of packet tunneling will not work correctly with NAT.

2.2.2 Covert Tunnel

A new distinction made by the Tentun project is that covert tunnels do not use a tunneling protocol and its associated data space. Instead, space is made or found within a packet of an existing data stream to send data from another data stream. Similar to a tunneling protocol, the tunnel client and server must be aware of which modifications were made to the legitimate packets. Legitimate is specified here with regards to the fact that the packets being used to tunnel are part of an already existing data stream or session.

To better illustrate this concept, let's consider covert tunneling using TCP options specifically. In figure 2, the TCP options field comes just before the data section and is separated by a padding field. The tunnel will use both the options and padding fields.

In figure 3, there are four endpoints. The tunnel will be using the TCP options field of the upstream provider and receiver's connection. The cropped section shows how two other endpoints can create a covert tunnel using packets from the original stream. The IP packet to be covertly tunneled will be split

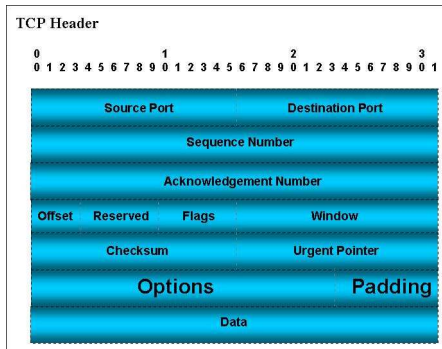


Figure 2: Figure 2 - TCP Header

into fragments of around 30 bytes to fit in the TCP options field. This obviously introduces a large latency in the covert tunnel. However, the tunnel is very difficult to detect by common means.

2.3 Tunnel scope

When a tunnel is established, it can be applied to the entire data stream or only on individual messages. We call the first approach stream tunneling and the second one message tunneling.

A data stream refers to a sequence of bytes between a client and a server. No special formatting is implied in this sequence. A message is an indivisible small sequence of bytes that must be transmitted together.

2.3.1 Stream Tunneling

Stream tunneling is a term that can be used when the tunnel client sends data to the server without considering each packet. For instance, the stream of data is separated into 1Kb blocks and each block is encapsulated within another protocol, possibly a tunneling protocol. The term stream tunneling is used here because the tunnel is only concerned with the stream of data.

2.3.2 Message Tunneling

Message tunneling refers to the fact that the messages produced by an application must be retransmitted while preserving their boundaries. For example, if an application generates two UDP messages, they must be sent to their destination without having their data combined.

Packet tunneling is a type of message tunneling where each message is a packet.

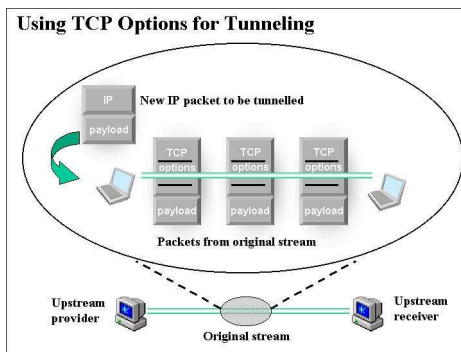


Figure 3: Figure 3 - TCP Options Covert Tunnel

3 Tunneling Implementations, protocol and tools

Throughout the internet, there are a number of open source and commercial tunneling implementations, tools or protocols. A few of the open source projects are listed here with a brief explanation of the project.

3.1 PPTP

One approach to tunneling is the Point-to-Point Tunneling Protocol (PPTP) developed by Microsoft and several other companies. PPTP keeps data secure,

even though part of the path can use a public communication channel.

The control channel is a standard TCP connection to port 1723 on the server. The data channel carrying the private network traffic uses IP protocol number 47 (GRE), a generic encapsulation protocol described in RFC1701. The transparent transmission of data over the data channel is achieved by negotiating a standard PPP connection over it, just as if it were a dialup connection directly from the client to the server.

3.2 L2TP

L2TP uses packet-switched network connections to make it possible for the endpoints to be located on different machines. The user has an L2 connection to an access concentrator, which then tunnels individual PPP frames to the NAS, so that the packets can be processed separately from the location of the circuit termination.

In the OSI network model, tunneling protocols can work on different layers. PPTP and L2TP use layer 2, which is the link layer.

3.3 IPSec

IPsec (Internet Protocol Security) is a developing standard for security at the network layer (layer 3 in OSI model). A big advantage of IPsec is that security arrangements (SA) can be handled without modifying applications or user systems.

To understand the scope of IPsec, SSL secures only one application socket; SSH secures only a login; PGP secures only a specified file or message. Whereas, IPsec encrypts everything between two hosts.

IPsec and L2TP are two concurrent approaches to secure tunneling. However, L2TP is at a lower level than IPsec. The advantage of being on a lower layer is that all higher layers are unaware of the lower layer's implementation. This means that existing security protocols such as RADIUS, which tracks, monitors and audits authentication, can be used seamlessly in L2TP installations. L2TP looks like any terminal server to RADIUS. Again, though, on the practical side, most vendors that offer IPsec products have also integrated support for RADIUS, SecureID, etc. into their products.

IPsec uses device assisted tunneling.

3.4 Httpunnel

Httpunnel creates a bidirectional virtual data connection tunnelled in HTTP requests. The HTTP requests can be sent via an HTTP proxy if so desired.

This can be useful for users behind restrictive firewalls. If WWW access is allowed through a HTTP proxy, it's possible to use httpunnel and, say, telnet to a computer outside the firewall.

3.5 ICMPChat

ICMPChat is an encrypted peer-to-peer chat that is based on the ICMP protocol. The data is encrypted using blowfish and sent by the payload of various ICMP packets.

3.6 VTun

VTun is the easiest way to create Virtual Tunnels over TCP/IP networks with traffic shaping, compression, and encryption. It supports IP, PPP, SLIP, Ethernet and other tunnel types. VTun is easily and highly configurable, it can be used for various network tasks.

3.7 Mobile IP

Mobile IP use tunneling to forward the message to the mobile user. Each IP packet to the original user is intercepted and put in another IP packet which is sent to the new address of the mobile user.

4 Conclusion

Finding covert tunnels is a hard problem. More work needs to be done at the IDS and firewall level. For example, a firewall should validate the length of the TCP options field with its actual length. We have presented a new tunneling technique that can use existing streams of data to hide a covert stream. We intend to pursue this avenue further to explore the potential security risks that tunneling can pose.